

The RSA Validation System (RSAVS)

June 14, 2004

Sharon S. Keller

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1	INTRODUCTION.....	2
2	SCOPE	2
3	CONFORMANCE	3
4	DEFINITIONS AND ABBREVIATIONS	3
4.1	DEFINITIONS.....	3
4.2	ABBREVIATIONS.....	3
5	DESIGN PHILOSOPHY OF THE RSA VALIDATION SYSTEM.....	4
6	RSAVS TESTS	4
6.1	CONFIGURATION INFORMATION	4
6.2	KEY GENERATION TEST	5
6.2	SIGNATURE GENERATION TEST.....	6
6.4	SIGNATURE VERIFICATION TEST	7
APPENDIX A	REFERENCES.....	9
APPENDIX B	EXAMPLE OF <i>REQUEST</i>, <i>FAX</i>, <i>RESPONSE</i>, AND <i>SAMPLE</i> FILES FOR RSA AS APPROVED IN FIPS186-2 AND SPECIFIED IN ANSI X9.31.....	10
B.1	Examples of <i>REQUEST</i> Files	10
B.1.1	KeyGenRSA.req	10
B.1.2	SigGenRSA.req	12
B.1.3	SigVerRSA.req	14
B.2	EXAMPLES OF <i>FAX</i> FILES	19
B.2.1	KeyGenRSA.fax	19
B.2.2	SigGenRSA.fax	22
B.2.3	SigVerRSA.fax	25
B.3	EXAMPLES OF <i>RESPONSE</i> FILES	35
B.3.1	KeyGenRSA.rsp	35
B.3.2	SigGenRSA.rsp.....	35
B.3.3	SigVerRSA.rsp	39
B.4	EXAMPLES OF <i>SAMPLE</i> FILES.....	44
B.4.1	KeyGenRSA.sam.....	44
B.4.2	SigGenRSA.sam	45
B.4.3	SigVerRSA.sam.....	47

1 Introduction

This document, *The RSA Validation System (RSAVS)* specifies the procedures involved in validating implementations of public key cryptography based on the RSA algorithm. This document deals with three variations of the RSA algorithm. They are:

- RSA algorithm specified in FIPS186-2, *Digital Signature Standard (DSS)*, January 27, 2000[1], and
- Two signature schemes with appendix specified in *Public Key Cryptography Standards(PKCS) #1 v2.1: RSA Cryptography Standard-2002* [2]. These two signature schemes with appendix are
 - RSASSA-PSS, and
 - RSASSA-PKCS1-v1_5.

The RSAVS only supports the RSA algorithm; that is, it only supports implementations where the public verification exponent e is odd. Both X9.31 and the PKCS #1 V2.1 documents support RSA. As specified in ANSI X9.31-1998, Section 4.1.1 (as pointed to by FIPS186-2), when the public key exponent is odd, the digital signature algorithm is commonly called RSA. When the public key exponent is even, the digital signature algorithm is commonly called Rabin-Williams.

The RSAVS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the RSAVS. Included are the specifications for testing the Key Generation, Signature Generation and Signature Verification components of the IUT. Note that the key generation validation test applies to the key generation procedure specified in ANSI X9.31.

This document defines the purpose, the design philosophy, and the high-level description of the validation process for RSA. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of RSA are presented. The requirements described include the specification of the data communicated between the IUT and the RSAVS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the RSAVS. Additionally, an appendix is also provided containing samples of input and output files for the RSAVS.

2 Scope

This document specifies the tests required to validate IUTs for conformance to the RSA as specified in [1] and [2]. When applied to IUTs that implement any of the three different algorithm variations of the RSA, the RSAVS provides testing to determine the correctness of the signature generation and verification components contained in the implementation. These two separate tests examine the signature generation and the signature verification algorithm components. For IUTs that implement the ANSI X9.31 variation of RSA, the RSAVS also provides testing to determine the correctness of the key generation components contained in the

implementation. Note that the PKCS document does not specify key generation procedures. In addition to determining conformance to the cryptographic specifications, the RSAVS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the RSA implementation.

3 Conformance

The successful completion of the tests contained within the RSAVS is required to be validated as conforming to the RSA. Testing for the cryptographic module in which the RSA is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*.[3]

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	MEANING
CMT laboratory	Cryptographic Module Testing laboratory that operates the RSAVS
RSA algorithm	The algorithm specified in the FIPS186-2, <i>Digital Signature Standard (DSS)</i> and the PKCS#1 v2.1 document.
PKCS	Public Key Cryptography Standards

4.2 Abbreviations

ABBREVIATION	MEANING
RSA	RSA algorithm specified in FIPS186-2
RSAVS	RSA Validation System
IUT	Implementation Under Test
PKCS	Public Key Cryptography Standards
RSASSA	RSA Signature Scheme with appendix
RSASSA-PKCS1_V1_5	PKCS # 1 Version 1.5 Signature Scheme with appendix
RSASSA-PSS	Probabilistic Signature Scheme with appendix

5 Design Philosophy Of The RSA Validation System

The RSAVS is designed to test conformance to RSA rather than provide a measure of a product's security. The validation tests are designed to assist in the detection of accidental implementation errors, and are not designed to detect intentional attempts to misrepresent conformance. Thus, validation should not be interpreted as an evaluation or endorsement of overall product security.

The RSAVS has the following design philosophy:

1. The RSAVS is designed to allow the testing of an IUT at locations remote to the RSAVS. The RSAVS and the IUT communicate data via *REQUEST* and *RESPONSE* files.
2. The testing performed within the RSAVS utilizes statistical sampling (i.e., only a small number of the possible cases are tested); hence, the successful validation of a device does not imply 100% conformance with the standard.

6 RSAVS Tests

The RSAVS provides conformance testing for three of the components of the algorithm, as well as testing for apparent implementation errors. The components tested are key generation, signature generation and signature validation.

6.1 Configuration Information

To initiate the validation process of the RSAVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of RSA. The vendor's implementation is referred to as the Implementation Under Test (IUT). The request for validation includes background information describing the IUT along with information needed by the RSAVS to perform the specific tests. More specifically, the request for validation includes:

1. Vendor Name;
2. Product Name;
3. Product Version;
4. Implementation in software, firmware, or hardware;
5. Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;
6. Brief description of the IUT or the product/product family in which the IUT is implemented by the vendor (2-3 sentences); and

7. The modulus size(s) supported by the IUT.
8. The SHA algorithms supported by the implementation.
9. For RSASSA-PSS implementations, a SALT length.
10. For IUT's implementing Key Generation, the fixed values of the public key, e , supported by the IUT. These include 3, 17, and/or 65,537.

6.2 Key Generation Test

An implementation of the RSA as specified in ANSI X9.31 may generate the key components used in the RSA algorithm's signature generation and verification processes. These key components include the public verification exponent, e , the private prime factors, p and q , the public modulus, n , and the calculation of the private signature exponent, d . This option tests the ability of an IUT to produce correct values for each of these components. To test key generation, the RSAVS supplies two sets of random numbers for each public key supported by the IUT. Each set consists of four random numbers used to generate the prime factors p and q . In the ANSI X9.31 standard, these random numbers are referred to as X_{p1} , X_{p2} , X_{q1} , and X_{q2} . The IUT calculates the corresponding key components and returns them to the RSAVS. The RSAVS compares the received results with its own stored results.

The RSAVS:

- A. Creates a *REQUEST* file (Filename: RSAKeyGen.req) containing:
 1. The Product Name
 2. The modulus size
 3. 2 groups of data for each public key exponent e supported consisting of
 - a. The public key supported,
 - b. Four random numbers: X_{p1} , X_{p2} , X_{q1} , and X_{q2} .

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

- D. Creates a *FAX* file (Filename: RSAKeyGen.fax) containing:
 1. The information from the *REQUEST* file and
 2. The private prime factor p
 4. The private prime factor q
 5. The value of the modulus n
 6. The value of the private signature exponent d .

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

The IUT:

- A. Uses the public key exponent e , and the four random numbers supplied in the *REQUEST* file to generate the key components.
- B. Creates a *RESPONSE* file (Filename: RSAKeyGen.rsp) containing:
 - 1. The information from the *REQUEST* file and
 - 2. The private prime factor p
 - 3. The private prime factor q
 - 4. The value of the modulus n
 - 5. The value of the private signature exponent d .

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the RSAVS.

The RSAVS:

- A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.
- B. Records PASS for this test if the results for all public key e , private prime factor p , private prime factor q , modulus n , and private signature exponent d match; otherwise, records FAIL.

6.2 Signature Generation Test

An implementation of the RSA may generate the digital signature. This option tests the ability of an IUT to produce correct signatures. To test signature generation, the RSAVS supplies ten messages to the IUT for each modulus size/SHA algorithm combination. The IUT generates the corresponding signatures and returns them to the RSAVS. The RSAVS validates the signatures by using the associated public key to verify the signature.

The RSAVS:

- A. Creates a *REQUEST* file (Filename: RSASigGen.req) and a *FAX* file (Filename: RSASigGen.fax) containing:
 - 1. The Product Name,
 - 2. The modulus size,
 - 3. Ten groups of data for each SHA algorithm supported consisting of
 - a. The SHA algorithm supported,
 - b. A message to be signed.

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

The IUT:

- A. Generates the signatures for the messages supplied in the *REQUEST* file.

- B. Creates a *RESPONSE* file (Filename: RSASigGen.rsp) containing:
1. The Product Name,
 2. The modulus, n ,
 3. The public key, e , corresponding to the private key, d , used to generate the signatures,
 4. Ten groups of data for each SHA algorithm supported consisting of
 - a. The SHA algorithm supported,
 - b. A message, Msg , and its corresponding signature value, s .

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the RSAVS.

The RSAVS:

- A. Uses the respective public keys to verify the signatures in the *RESPONSE* file.
- B. Records PASS for this test if all conditions are met; otherwise, records FAIL.

6.4 Signature Verification Test

This option tests the ability of the IUT to recognize valid and invalid signatures. For each modulus size/SHA algorithm selected, the RSAVS generates a modulus and three associated key pairs, (d, e) . Each private key d is used to sign four pseudorandom messages each of 1024 bits. Some of the public keys, e , messages or signatures are altered so that signature verification should fail. The modulus, SHA algorithm, public key e values, messages, and signatures are forwarded to the IUT. The IUT then attempts to verify the signatures and returns the results to the RSAVS, which compares the received results with its own stored results.

The RSAVS:

- A. Generates 3 groups of data for each supported modulus size. Each group consists of a modulus and 4 sets of data for each supported SHA algorithm. Each set of data contains
 1. The SHA algorithm supported,
 2. A public/private key pair that is consistent with the modulus,
 3. A pseudorandom message and
 4. A signature for the message using the private key.

For the efficiency of the tool, a modulus will sometimes be used for more than one group of data when it is consistent with more than one public key. Therefore when loading the modulus for each group of data, the modulus specified for this group may be the same as that specified for the previous group.

- B. Alters the public key e , the message or the signature for three fourths of the public key/message/signature sets such that the message verification fails.
- C. Creates a *REQUEST* file (Filename: RSASigVer.req) containing:
 - 1. The Product Name;
 - 2. For each modulus size supported:
 - a. The modulus n for the supported modulus size,
 - b. 3 groups of data consisting of 4 sets of data for each supported SHA algorithm. Each set of data contains:
 - i. The SHA algorithm supported,
 - ii. The information from step B, including:
 - 1. A public key e corresponding to the private key used to sign the messages,
 - 2. The pseudorandom message and
 - 3. The signature.

Note: The CMT laboratory sends the *REQUEST* file to the IUT.

- D. Creates a *FAX* file (Filename: RSASigVer.fax) containing:
 - 1. The information from the *REQUEST* file and
 - 2. An indication of whether the signature verification process should pass or fail, for each public key/message/signature set.

The IUT:

- A. Attempts to verify the signatures for the messages supplied in the *REQUEST* file using the corresponding modulus n , SHA algorithm and the public key e .
- B. Creates a *RESPONSE* file (Filename: RSASigVer.rsp) containing:
 - 1. The information from the *REQUEST* file and
 - 2. An indication of whether the signature verification passed or failed for each public key/message/signature set.

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the RSAVS.

The RSAVS:

- A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.
- B. Records PASS for this test if the results for all public key/message/signature sets match; otherwise, records FAIL.

Appendix A References

- [1] FIPS186-2, Digital Signature Standard (DSS), January 27, 2000.
- [2] PKCS#1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002.
- [3] *Security Requirements for Cryptographic Modules*, FIPS Publication 140-2, National Institute of Standards and Technology, May 2001.

Appendix B Example of *REQUEST*, *FAX*, *RESPONSE*, and *SAMPLE* Files for RSA as approved in FIPS186-2 and specified in ANSI X9.31

The following examples contain values that are longer than one line. These values should be on one line. For example:

P =
f73accd5721dad7307a70cd5c00e3d028e323781e362e17c327b239077f53cf0496b14a1fa57e
0bc18fd308fcc6c8bd2c5fcbb457bc5146cb1128f92fc9c7b3b8608e40c56c343fd0adb47c6a5d
9f55065ae42e4aab900c70fcc19cfdf9b7c19ca5118dbfc5ed4f26dd9a7dc010580c49ed2cf5
12b7239b15a1eddca82e45

is the character sequence 'P', <space>, '=', <space>, 'f', '7', '3', ..., '4', '5' followed by a <newline>.

Note that these are not complete files. They are only examples of what the files would look like.

Please refer to RSAExample.zip for examples of complete files for the ANSI X9.31 RSA algorithm and for the RSASSA-PSS and RSASSA-PKCS1-v1_5 signature schemes with appendix specified in PKCS#1 v2.1.

B.1 Examples of *REQUEST* Files

B.1.1 KeyGenRSA.req

[2 for each public key: 3, 17, and 65,537]

[mod = 1536]

```
e =
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
xp1 = 1e64c1af460dff8842c22b64d0
xp2 = 1e948edcedba84039c81f2ac0c
Xp =
c8c67df894c882045ede26a9008ab09ea0672077d7bc71d412511cd93981ddde8f91b967da4040
56c39f105f7f239abdaff92923859920f6299e82b95bd5b8c959948f4a034d81613d6235a3953b
49ce26974eb7bb1f14843841281b363b9cdb
xq1 = 1f3df0f017ddd05611a97b6adb
xq2 = 143edd7b22d828913abf24ca4d
Xq =
f15147d0e7c04a1e3f37adde802cdc610999bf7ab0088434aaeda0c0ab3910b14d2ce56cb66bff
d97552195fae8b061077e03920814d8b9cfb5a3958b3a82c2a7fc97e55db543948d33962892453
36ec9e3cb308cc655aebd766340da8921383
```

[2 for each public key: 3, 17, and 65,537]

[DO FOR EACH MOD SIZE]

B.1.2 SigGenRSA.req

```
# CAVS 3.2
# "SigGen RSA (X9.31)" information for "testshas"
# Mod sizes selected: 1024 1536 2048 3072 4096
# SHA Algorithm selected:SHA1 SHA256 SHA384 SHA512
# Generated on Wed Apr 28 08:34:41 2004

[mod = 1024]

SHAAlg = SHA1
Msg =
940562ae51c1990882b27e7335fbb8c871db97e625d5a8f95f0f86fcbe9f27c2a2e269fc29f67
6616eeda2f9718987e4c5704fcc6475dc055478fce7c224fdda3ef665e0c354db90853fbda6b1f
fbec3dcec164b2143e425c8bb293a059c23b670ecaf115cb748bbfa98a7c9958089bad077626d3
847406dda5975412e6731d

SHAAlg = SHA1
Msg =
61a908f8f1da17288dc06d4611df5503b79385cf80eca04ed6bbff056fedb15a7418c0bbe354b6
1d324c60a83595d2b0413eabe892a89bd2ea97227a7b8a9a64074877c346bcc880214099bc2
2912efbd94f9f8a51125d43249222e72e0976261b478e1b9647cd80b10d20c0f60100839c86c7b
8c0a2edcb3fc654f4e8bd9
[FOR 10 ITERATIONS]

.
.

SHAAlg = SHA256
Msg =
dbc0695509efe8ed418c85fc9f106073638adf6e469fa35d0fcfb6161fa17f9d7223fdd537245
2a927ff7c004cf5ae8db98543b62c9a9bf914e2952de90274c553c2c60eb46edc3102d7b908380
ba6c6aa11466a2c96e20544c5b34c91f90d17f9799a57c73ca00e21d7736c42d6845382f87b7ad
a6dccca7f51bbcfc9ac3cd0

SHAAlg = SHA256
Msg =
26bc9efb0bd3467b5f95037fe881e3284c79d8f5237e699e4fbca84090c664bb53229f58cb0842
b0436710c9b329d98191b8f030e9c1df89b03858c1569c6ff49a7c07c4a23a8a434b0fde13be4f
94cb44ee629d5b44d336090d3de670b4f401a3d1bd85b8f085fc1e9453a4317b7cce0c2416849e
8fae6e01443ef7069659ab
[FOR 10 ITERATIONS]

.
.

SHAAlg = SHA384
Msg =
06ac89b655b1e0bd5e7f0004cf9aa765ecc4e3c4d72b77e70ee28088358e58a91662bb28e64d28
93c9718a3d99d81892e3627aa733022d16922a28084c84c093f7e3c947b079fe03a84aa30de0e0
68623041914b8e1e54318d4d82b2247a6af5b119fb3a3d9e28b502f1919c2a1c5f7fda476fd86f
e7b4e30832d6af44d61f75

SHAAlg = SHA384
Msg =
bc55612bb063b5f05da74569b3a39cc9abb99f2c7c93651f12c24863bca53cee258d6033851ed3
a318f9c974d098dea14778aa32f77e95bdad94ec2d3b9335c26d65c0593f6b7fee4d1c175f8bef
```

```

3772367b291a0bdda7f6b65bef1b8d471a137e25a925461061d7e45959b24e725145620a456d1f
7a42d3156079b51a8992ea
[FOR 10 ITERATIONS]

.

.

.

SHAAlg = SHA512
Msg =
82026ab0d0445fdc66733e39a205f50a8e6e7a80619369d24982ef906760e6341ba241fa8381ab
f13ea76b52762f8b11dcc39b376486d5fd831cc37afcfcd9c3581a80bd673981e7ab6b3333dc
d43e7ff4dd77179bd7fb7be3e17b63350bd6018cbb86e050b3e7e60683e1f619b73b4c2e764b
74eecdbb00f4018d7ce924

SHAAlg = SHA512
Msg =
3d52ea07bla428f4385d4ed0fc53e8c8fed02e074fef63782492d4561d16665dff32574e2791
e8d232e7bb167052493dc33b271c032b88a6a0e002a789a195b64cce9647ebbd7ba5dbe2be3b4
87a825ffb04d16d095d70c716687d5cb7b25a886e7455c724fc9d826fda7cbe730ed9dcbb602a5
1dbd2e9c04e75c51609c1f
[FOR 10 ITERATIONS]

.

.

.

[mod = 1536]

SHAAlg = SHA1
Msg =
9e2911b0fe05cadbbc509e8685d3bc7229ed9c8cf192ee123494d0f625214387ef8ee04e0452c7
36b156d9f1f76c2796961ca98f6aa734b2fa46a6ae4fb364a902ea06279931502e145664a8c238
37085988423046d17a9242b68196267c1e3c00c8eb0366a94d090ad8a9738f3fd50f9bb9cd382e
19a8fb326dca7853845a8a

SHAAlg = SHA1
Msg =
36cb62c9796e73e4847cf694077aabac5aeb56ef07c665238827d26852454bc812a4910f368c1e
ddafb90af9f5542fe84c228cbf47fce7358a7fb6fc30440523cd200d3ba934a91ea0531660df2
a7895062e660520f0f95e019116c03fbcd543cbe78bc0e6d4027bdc83283563c5bb6ef132d7b30
e9256c125fab5fe7133af6
[FOR 10 ITERATIONS]

.

.

.

SHAAlg = SHA256
Msg =
212850d3481b80676096014cb7c5202d98d03495b7af2360ce4027708996d2a1a09450fe42bbcd
35ba41d6484d63248661fc16bae5559ea2ec3f20a29b5f76ba46ca37b6d6d239a46e1f548b4e01
695af704a31f8fc5fd79880505507dfcdfc35cef8c62ca056d9ec357027d8146f893a011632d7f
a748cfe9b95ee1f51415a5

SHAAlg = SHA256
Msg =
924472ca0a9d5e5c9199fb110e8e180c68d96abca675d30c55cce3da2acca0abe93e829d8553c2
9d8980d5d9b68c3378eee85c89513e1d3aa54c0f4bfb362c28f8bd8dff17a1452620515ac8166
fe14271e183b8f8c1992babfa7ba1189183b4dd47c7a9439dde5e4cf571b5410146da24a5b8a9
87507a5bcbb10eb6b54a80
[FOR 10 ITERATIONS PER SHAAlg SUPPORTED]
.
```

[mod = 2048]
[SAME FORMAT AS ABOVE]

[mod = 3072]
[SAME FORMAT AS ABOVE]

[mod = 4096]
[SAME FORMAT AS ABOVE]

B.1.3 SigVerRSA.req

B.2 Examples of *FAX* Files

B.2.1 KeyGenRSA.fax

```
# CAVS 3.2
# "KeyGen RSA (X9.31)" information for "keygen1half"
# Mod sizes selected: 1024 1536 2048
# Public Keys selected:3 17 65537
# Generated on Mon May 24 15:27:21 2004

[mod = 1024]
```

e =

35292f4f4532aac696ac4f8eb0e374be5add7a8015c72e84cba71e153d8970bfbea313cb297c6f
9f981f81e46996276389c5d69c047ae4e8bc9a34ebb4bcac67b98f0078b60801155bc5a261ecc2
4293fb2ce7f2e9171be7df

d =
278d07b248b087dc63f1268de50066736a98944f923a59cc05e0c588f4d7459c102044984e0eeff
b386dd37e0ddc721191cb7ed1d7b3e1fb9cf946aae4bdd16219effae4f6846c6114a61169279fe
baddf7762fdb73ff6cd7e36d0528cf66239cf4f17a3fb13637cd27404865298fe3918260890d12
3384aa6831b91404eabce3

[2 for each public key: 3, 17, and 65,537]

[mod = 1536]

[2 for each public key: 3, 17, and 65,537]

[DO FOR EACH MOD SIZE]

B.2.2 SigGenRSA.fax

```
# CAVS 3.2
# "SigGen RSA (X9.31)" information for "testshas"
# Mod sizes selected: 1024 1536 2048 3072 4096
# SHA Algorithm selected:SHA1 SHA256 SHA384 SHA512
# Generated on Wed Apr 28 08:34:41 2004

[mod = 1024]

SHAAlg = SHA1
```

```

Msg =
940562ae51c1990882b27e7335fbb8c871db97e625d5a8f95f0f86fcbef9f27c2a2e269fc29f67
6616eeda2f9718987e4c5704fcc6475dc055478fce7c224fdda3ef665e0c354db90853fbda6b1f
fbecc3dcec164b2143e425c8bb293a059c23b670ecaf115cb748bbfa98a7c9958089bad077626d3
847406ddaa5975412e6731d

SHAAlg = SHA1
Msg =
61a908f8f1da17288dc06d4611df5503b79385cf80eca04ed6bbff056fedb15a7418c0bbe354b6
1d324c60a83595d2b0413eabe892a89bd2ea97227a7b8a9a64074877c346bcc880214099bc2
2912efbd94f9f8a51125d43249222e72e0976261b478e1b9647cd80b10d20c0f60100839c86c7b
8c0a2edcb3fc654f4e8bd9
[FOR 10 ITERATIONS]
.

.

.

SHAAlg = SHA256
Msg =
dbc0695509efe8ed418c85fc9f106073638adf6e469fa35d0fcfbb6161fa17f9d7223fdd537245
2a927ff7c004cf5ae8db98543b62c9a9bf914e2952de90274c553c2c60eb46edc3102d7b908380
ba6c6aa11466a2c96e20544c5b34c91f90d17f9799a57c73ca00e21d7736c42d6845382f87b7ad
a6dcca7f51bbcf9ac3cd0

SHAAlg = SHA256
Msg =
26bc9efb0bd3467b5f95037fe881e3284c79d8f5237e699e4fbca84090c664bb53229f58cb0842
b0436710c9b329d98191b8f030e9c1df89b03858c1569c6ff49a7c07c4a23a8a434b0fde13be4f
94cb44ee629d5b44d336090d3de670b4f401a3d1bd85b8f085fc1e9453a4317b7cce0c2416849e
8fae6e01443ef7069659ab
[FOR 10 ITERATIONS]
.

.

.

SHAAlg = SHA384
Msg =
06ac89b655b1e0bd5e7f0004cf9aa765ecc4e3c4d72b77e70ee28088358e58a91662bb28e64d28
93c9718a3d99d81892e3627aa733022d16922a28084c84c093f7e3c947b079fe03a84aa30de0e0
68623041914b8e1e54318d4d82b2247a6af5b119fb3a3d9e28b502f1919c2a1c5f7fd476fd86f
e7b4e30832d6af44d61f75

SHAAlg = SHA384
Msg =
bc55612bb063b5f05da74569b3a39cc9abb99f2c7c93651f12c24863bca53cee258d6033851ed3
a318f9c974d098dea14778aa32f77e95bdad94ec2d3b9335c26d65c0593f6b7fee4d1c175f8bef
3772367b291a0bdda7f6b65bef1b8d471a137e25a925461061d7e45959b24e725145620a456d1f
7a42d3156079b51a8992ea
[FOR 10 ITERATIONS]
.

.

.

SHAAlg = SHA512
Msg =
82026ab0d0445fdc66733e39a205f50a8e6e7a80619369d24982ef906760e6341ba241fa8381ab
f13ea76b52762f8b111dcc39b376486d5fd831cc37afcfcde9c3581a80bd673981e7ab6b3333dc
d43e7ff4dd77179bd7fb7be3e17b63350bd6018cbb86e050b3e7e60683e1f619b73b4c2e764b
74eecdbb00f4018d7ce924

SHAAlg = SHA512

```

```

Msg =
3d52ea07b1a428f4385d4ed0fc53e8c8fed02e074fefd63782492d4561d16665dff32574e2791
e8d232e7bb167052493dc33b271c032b88a6a0e002a789a195b64cce9647ebbd7ba5dbe2be3b4
87a825ffb04d16d095d70c716687d5cb7b25a886e7455c724fc9d826fda7cbe730ed9dcbb602a5
1dbd2e9c04e75c51609c1f
[FOR 10 ITERATIONS]

.

.

[mod = 1536]

SHAAlg = SHA1
Msg =
9e2911b0fe05cadbbc509e8685d3bc7229ed9c8cf192ee123494d0f625214387ef8ee04e0452c7
36b156d9f1f76c2796961ca98f6aa734b2fa46a6ae4fb364a902ea06279931502e145664a8c238
37085988423046d17a9242b68196267c1e3c00c8eb0366a94d090ad8a9738f3fd50f9bb9cd382e
19a8fb326dca7853845a8a

SHAAlg = SHA1
Msg =
36cb62c9796e73e4847cf694077aabac5aeb56ef07c665238827d26852454bc812a4910f368c1e
ddaf90af9f5542fe84c228cbf47fce7358a7fb6fc30440523cd200d3ba934a91ea0531660df2
a7895062e660520f0f95e019116c03fbcd543cbe78bc0e6d4027bdc83283563c5bb6ef132d7b30
e9256c125fab5fe7133af6
[FOR 10 ITERATIONS]

.

.

SHAAlg = SHA256
Msg =
212850d3481b80676096014cb7c5202d98d03495b7af2360ce4027708996d2a1a09450fe42bbcd
35ba41d6484d63248661fc16bae5559ea2ec3f20a29b5f76ba46ca37b6d6d239a46e1f548b4e01
695af704a31f8fc5fd79880505507dfcd9c35cef8c62ca056d9ec357027d8146f893a011632d7f
a748cfe9b95ee1f51415a5

SHAAlg = SHA256
Msg =
924472ca0a9d5e5c9199fb110e8e180c68d96abca675d30c55cce3da2acca0abe93e829d8553c2
9d8980d5d9b68c3378eee85c89513e1d3aa54c0f4bf9f362c28f8bd8dff17a1452620515ac8166
fe14271e183b8f8c1992babfa7ba1189183b4dd47c7a9439dde5e4cfcc571b5410146da24a5b8a9
87507a5bccbb10eb6b54a80
[FOR 10 ITERATIONS PER SHAAlg SUPPORTED]

.

.

[mod = 2048]
[SAME FORMAT AS ABOVE]

.

.

[mod = 3072]
[SAME FORMAT AS ABOVE]

.
.
```

[mod = 4096]
[SAME FORMAT AS ABOVE]

B.2.3 SigVerRSA.fax

```
# CAVS 3.2

# "SigVer RSA (X9.31)" information for "testshas"

# Mod sizes selected: 1024 1536

# SHA Algorithm selected:SHA1 SHA256 SHA384 SHA512

# Generated on Wed Apr 28 08:35:11 2004
```

[mod = 1024]

```
n =
9ec4d483330916b69eee4e9b7614eafc4fbf60e74b5127a3ff5bd9d48c7ecf8418d94d1e60388b
b68546f8bc92deb1974b9def6748fbb4ec93029ea8b7bea36f61c5c6aeefd512a0f765846fad5
edacb08c3d75cf1d43b48b394c94323c3f3e9ba6612f93fe2900134217433afb088b5ca33fc4e6
b270194df077d2b6592743
```

```
p =  
c3cd741d76dff6aebc64a234d077bc303c4b361ca9b52607f6ea787f8789e0b3e0dc13d9725f9a  
7eb55dd8dc6335dd9603bdba29320ff371cc72593f78433c07
```

```
q =  
cf94a874e82decba20a950449d225817a1e4ec0ee8c658cf4bb97fdc7a4d1b0d06822228b5764  
dc99b9e1b9ea43bb3fea530fc802124b73c67d523f8e24a3e5
```

~~SHA1q~~ = SHA1

```
d =  
1a76236b332c2e73c527b7c493ae272a0d4a90268c8d869b5539f9a36cbfcfd40aecee22fbab417  
4916367eca187a72ee8c9a529136d49e276dd5c51c1e9fc5e7a265f1af6d27e7a03311664cecdf  
4ee6230f59e1b4c5a0cf754334ae7b2ceccfc65b1dde61319b76070f722795929e3d1868f89d  
c2b2ddf480220df2a8368f
```


SHA1q = SHA256


```

Result = F (2)

n =

9ec4d483330916b69eee4e9b7614eafc4fbf60e74b5127a3ff5bd9d48c7ecf8418d94d1e60388b
b68546f8bc92deb1974b9def6748fb4ec93029ea8b7bea36f61c5c6aeedfd512a0f765846fad5
edacb08c3d75cf1d43b48b394c94323c3f3e9ba6612f93fe2900134217433afb088b5ca33fc4e6
b270194df077d2b6592743

p =
c3cd741d76dff6aebc64a234d077bc303c4b361ca9b52607f6ea787f8789e0b3e0dc13d9725f9a
7eb55dd8dc6335dd9603bdba29320ff371cc72593f78433c07

q =
cf94a874e82decba20a950449d225817a1e4ec0ee8c658cf4bb97fdc7a4d1b0d06822228b5764
dc99b9e1b9ea43bb3fea530fc802124b73c67d523f8e24a3e5

```

[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]

```

n =
b65fd92021a7aa326a9d2234797a90b7272a3251b5a2c3b119878ab71b60016fe0070f6395019d
149b35e82d408b77a9529252d6954f5e66d649b7c4ea1704114a130e99f93357b8253a2a51ee7c
c615862904c7b958f47d2cf6343060a57764fbab6b66b25b1e8f2cebf05c1ff23fe8cccd15d0c5
f8aae94f1fcf1b1c7ad1fd

p =
beba2012be61ef90bb40112c239c8773999aaf61a530d5ee02a4a9b865e7dde6edbaaa86058ad
02472833db839b8526b95657d8bf66499ad56ccb36af545ecd

q =
f4c9ef7a921559928c9ca1664800ed5830531aa127375195c143db28b3a936b2881668ef96dde2
edd497bc28d2fd06b6784129e2fb6fee20fa8d03d9b5ddff1

```

[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]

```

[mod = 1536]

n =
e0be7bc34325ce6c764fd6a7b09ac65a3c0d25330c978156127709050bbef1b6cc1d1473aca8d2
95606b40e6d39bc41d63294c7826c7a89eda8cf94c3fc55d2cbd172dbeef97bba7090a0ee04c7
1034fedcc63185bc264edcea6baf007d6525322ff5cb9d709ecae8179a7e5bb7784a17cfe82810
265947c8db4da2d9f4b9c3c89d045df163f0780dd67bec9d6c0325e8ac90bd069e04db4d1ab01f
3f7797b0d33e3373911275aa4e9a986dcdd25f3950a80d03ffac15443c639148be5d04e9

p =
f23fea6375f9b0736607247b6028db512f05a21bb40c07bf8b7b1c9becbb7777cff07db754cee1
d561ae670f8de1ee5d3d44ada60c14bd337104d7649c8d60a2a086a294d7f6834e0477810ca518
e0b12c6e91a4d4e00f3945b780669a1f59f1

q =
ed803286be2d8e956fbe2b0b641ba624ba78b1db4fcdb6ff8486be89417c922d3a251f2f71071f
78fc73f6e9096fdaef1ff43758a75a5265aa102c20dc90a9efda92f16ab083a01ecf0ebe3b4e0b
afa3bce6d153a975182d7910748bcfb1a279

[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]

```

B.3 Examples of *RESPONSE* Files

B.3.1 KeyGenRSA.rsp

Same as FAX file

B.3.2 SigGenRSA.rsp

```
# `SigGen931'
[mod = 1024]
```

```

n =
b82d9d45ddb8e39d1a43b0355f0037f6295ae7d1e056987eb316a60fa044f1c0356094be60b9c8
dad3cf37576c246060dd131bf17ed036beb6f06749c7be87d1c287fec1f01e51eda76cdb68ef6f
e0fa562d8e76cc8709243b3cdb87e4f751fa4d47b371fbb97c59c27f5450339fb2f6d93f6f699d
37781bcb75712b80d5b6a9

e = 3

SHAAlg = SHA1
Msg =
940562ae51c1990882b27e7335fbb8c871db97e625d5a8f95f0f86fcbef9f27c2a2e269fc29f67
6616eeda2f9718987e4c5704fcc6475dc055478fce7c224fdda3ef665e0c354db90853fbda6b1f
fbecc3dcec164b2143e425c8bb293a059c23b670ecaf115cb748bbfa98a7c9958089bad077626d3
847406dda5975412e6731d
S =
5b91b243f79ef2b894f2a32610853e66bc538750df1336feb1f9fd50088b66426020e120fadd0
4cc6ad53c6a4bb605b50a0a6671c6ae45dde27b9ae30181dc8c1e0598c06cf06b8c615de499e48
7d47ebfa37b1ffabbcfd765164682f3f0782afa6e5b2564168e8a03319898e0a703bb1925f714a
8522c4066963c2ab280760

SHAAlg = SHA1
Msg =
61a908f8f1da17288dc06d4611df5503b79385cf80eca04ed6bbff056fedb15a7418c0bbe354b6
1d324c60a83595d2b0413eabe892a89bd2ea97227a7b8a9a64074877c346bccceeb880214099bc2
2912efbd94f9f8a51125d43249222e72e0976261b478e1b9647cd80b10d20c0f60100839c86c7b
8c0a2edcb3fc654f4e8bd9
S =
1ff8dbc3e530f9f37ebd5b85a03faad0282dd523589b80cd7854429a11dc9e52f4ab613ff60d55
1cf4704707ca32e57f3c8005c42ce0cba10e244835a5e3411ef2eae0512e941e768a2e8cfee58
d58ec5ecc155e057f81f6520e66035fb55b0e91d12c66bdb897aad9aa7372fa7aeecc48e47e491d
77683a5be665ae477eaal8

[FOR 10 ITERATIONS]
.
.
.

SHAAlg = SHA256
Msg =
dbc0695509efe8ed418c85fc9f106073638adf6e469fa35d0fcfb6161fa17f9d7223fdd537245
2a927ff7c004cf5ae8db98543b62c9a9bf914e2952de90274c553c2c60eb46edc3102d7b908380
ba6c6aa11466a2c96e20544c5b34c91f90d17f9799a57c73ca00e21d7736c42d6845382f87b7ad
a6dccaf7f51bbcfc9ac3cd0
S =
26da2dcbeafbae58b16822a8482d7fb58611203fead935f74c99c5f71574ea3b6dbe6fb0d3e65c
261b162822c1a3688eedaac951f6be3ccda01d6d9f3b1fecc77c67b5fc5c014750e14e0b664a4d
89a14adf39f099cea770c3082e58566fafad2476a844dcfd3d617ac610adb8183ad4fee9eb2de4
0b3761163b2f9c6225fe68

SHAAlg = SHA256
Msg =
26bc9efb0bd3467b5f95037fe881e3284c79d8f5237e699e4fbca84090c664bb53229f58cb0842
b0436710c9b329d98191b8f030e9c1df89b03858c1569c6ff49a7c07c4a23a8a434b0fde13be4f
94cb44ee629d5b44d336090d3de670b4f401a3d1bd85b8f085fc1e9453a4317b7cce0c2416849e
8fae6e01443ef7069659ab
S =
5ab49af3f2dca02f5a15af065a24aa40e32d15f6a4a59c90a19c730402ef982a58cdde9e4d500
2baa7c69605eb6e4e98fbf2ea62083011a1477c380473bb23e24788d0e1cea71bded71251e3bc2
d0134bb5972e6512432ac4ff525dae18ee44197756d44450123ea3d3354e848efb6d838a024690
4e90fdc7fe4a0525918a4b

```

[FOR 10 ITERATIONS]

```
.
```

```
.
```

```
.
```

```
SHAAlg = SHA384
Msg =
6ac89b655b1e0bd5e7f0004cf9aa765ecc4e3c4d72b77e70ee28088358e58a91662bb28e64d289
3c9718a3d99d81892e3627aa733022d16922a28084c84c093f7e3c947b079fe03a84aa30de0e06
8623041914b8e1e54318d4d82b2247a6af5b119fb3a3d9e28b502f1919c2a1c5f7fd476fd86fe
7b4e30832d6af44d61f75
S =
50f3717def28d91da669c7b2ce0097443247c428db28c97d0d73c5dbd5017333d634868c42267f
2e62e75ef47a2be001ca16a17ec61b0ecf379782a4a5276b96346c2ba5c79ace966034eccfaa40
f5536c5c519ed8d3dc4e451e5fab2d0776e85dbdd99702fb794a95e40ce82c8749a46dc5431ce1
9ec7e9ebfb494548f8a13c
```

```
SHAAlg = SHA384
Msg =
bc55612bb063b5f05da74569b3a39cc9abb99f2c7c93651f12c24863bca53cee258d6033851ed3
a318f9c974d098dea14778aa32f77e95bdad94ec2d3b9335c26d65c0593f6b7fee4d1c175f8bef
3772367b291a0bdda7f6b65bef1b8d471a137e25a925461061d7e45959b24e725145620a456d1f
7a42d3156079b51a8992ea
S =
59fd61dfa126944bfe719bd4d48184595e483938b344a249d14d0e15c1e5cfa2bda6b3a8b40e25
621878ad4860cb217a3221de19e2d85819dadf3d650b01450ec5d0e1911f9a1e1b21140b16f753
2c1c50026f60888d4e5633cd255958a83bfeaaab48869cc74aba38ba022572ef1e46dc734852c0
22f8a43067e785ae91b4c2
```

[FOR 10 ITERATIONS]

```
.
```

```
.
```

```
.
```

```
SHAAlg = SHA512
Msg =
82026ab0d0445fdc66733e39a205f50a8e6e7a80619369d24982ef906760e6341ba241fa8381ab
f13ea76b52762f8b111dcc39b376486d5fd831cc37afcfcd9c3581a80bd673981e7ab6b3333dc
d43e7ff4dd77179bd7fb7be3e17b63350bd6018cbb86e050b3e7e60683e1f619b73b4c2e764b
74eecdbb00f4018d7ce924
S =
41c2b55ec08ab5383dd7bffe897f2a065633ef0c30af6919eaee9375e8fc146212509d93b72fbe
574d66add42b9d91b3153106e1bf5718d61d4b9278bf617f9ada4904d1b5e5a16a65e2359d32bd
659af03b13d1f52db7eb60561361b8955f859a71dcf3c37977aefff9bf706953ac2202f7e72a3
7c5b3473a590aac63a8b02
```

```
SHAAlg = SHA512
Msg =
3d52ea07b1a428f4385d4ed0fc53e8c8fed02e074fef63782492d4561d16665dff32574e2791
e8d232e7bb167052493dc33b271c032b88a6a0e002a789a195b64cce9647ebbd7ba5dbe2be3b4
87a825ffb04d16d095d70c716687d5cb7b25a886e7455c724fc9d826fda7cbe730ed9dcbb602a5
1dbd2e9c04e75c51609c1f
S =
1f40f049a6fcabe227d3f96389bbbcf6bcd28d941d8d9a3908d4819af12f3c3e8db07cdcb351b4
a3ede67e71e79f52a2bf246457d4eb73f6822d002e63ea275700f4989d072a56d9cf39ef8d2dc
c5ba32fb4f7569c331a9d72139a0f7da5f18476e5f98b525a23009f9c37e2ffceb33b2968eba35
deb240f859b674f6b58604
```

```

[FOR 10 ITERATIONS]
.

.

.

[mod = 1536]

n =
9b344a18fe6073a44083e8f32cc8a3b93b4e1399839d07581b42d2174502784bbd7189fe57b8ba
295a9bbf6474681ea6c8c50910edf0c2d18b4b30762af0038d89962b29e3f7f3bf1a23376db8f8
94f08b3809827f133547c2c3c49be46aa8f906dd61315ec23d0092ffe0d8872c81feaf1aea5ed0
282bddfae9bd665a5dcab638abbb14093cc817be07f1fb5e155fea80c41e22e584ec6eb10a581
20bdb864c895b6660628f9189926c542055f055ed6865525f2c11aaaa518409dal03a75

e = 3

SHAAlg = SHA1
Msg =
9e2911b0fe05cadbbc509e8685d3bc7229ed9c8cf192ee123494d0f625214387ef8ee04e0452c7
36b156d9f1f76c2796961ca98f6aa734b2fa46a6ae4fb364a902ea06279931502e145664a8c238
37085988423046d17a9242b68196267c1e3c00c8eb0366a94d090ad8a9738f3fd50f9bb9cd382e
19a8fb326dca7853845a8a
S =
28ba1761b6fe24cedb60a819446bc132618587ef360401b69f82891cb97e6f7f95c023834099a6
5d056e148790d9663d905a0b04d7537cb9c11c619d769579af608392003b8b7ceed6a29aed8a5d
081525dccd4e809163bef7c7472de82e4e11d1a1cdcd6a5c27bc56bd879de2d15f06b266c1e8e2
87e27f81a418d8c46528cc99925991a25f2fc6f9fc4f03f7c1335450ba62518ece395e72477e0
2c7ba2bff97b37a872117423ef969b47be41aeb9b7329c49840ffcdc00cbb5f2cfb70551

SHAAlg = SHA1
Msg =
36cb62c9796e73e4847cf694077aabac5aeb56ef07c665238827d26852454bc812a4910f368c1e
ddafb90af9f5542fe84c228cbf47fce7358a7fb6fc30440523cd200d3ba934a91ea0531660df2
a7895062e660520f0f95e019116c03fbcd543cbe78bc0e6d4027bdc83283563c5bb6ef132d7b30
e9256c125fab5fe7133af6
S =
20acbf8dfa928c75725eb274bb66c8ec74ca0c2a158687720d65ccf2e8bb4302029074d22029bf
729cb2ffc3118f2312e85690686c71bce50af565ca55db4bd9431e4caa0af18917f8f2592f0566
ca7cc81646b3515dce1545de09a4f390a4cccec94259bbf08861d7f3355085494f6efe8d684a98f
a3c60592e120b4ab2b785caaec2dea9e55f6a18364f578fd077db51f65356a3bb06fd9328d19df
425341ef640ca8524f1ee1d1d49c29cal1d6b735e6d26b70e7b0a41e833199231a250e3d4

[FOR 10 ITERATIONS]
.

.

.

SHAAlg = SHA256
Msg =
212850d3481b80676096014cb7c5202d98d03495b7af2360ce4027708996d2a1a09450fe42bbcd
35ba41d6484d63248661fc16bae5559ea2ec3f20a29b5f76ba46ca37b6d6d239a46e1f548b4e01
695af704a31f8fc5fd79880505507dfcdcf35cef8c62ca056d9ec357027d8146f893a011632d7f
a748cfe9b95ee1f51415a5
S =
37a83e329d8c13599f1a5b5e9c642a289ea215a903e7085d4e567f04f90f09d8cb85cc6cbe485b
7507845ce36cbc9e52fd90a9a09b2dafbee956d64a6f828d5b3f0fbfb5323fa1d4c99c07b790fc
be520014db7d719732439fa8091aaf4db30acd9d2bdgee75d7a85db727e79b2b3fc1787d954d5e
e9045cd87c8ad1d8bf58c6200a191dbc0184f74200607e02086feabbfe000d5abf0efa2eebed7a
20e2b599f6c9cc7a22861e6fc441392e0eac57720d6c9eb7002446262ae68f930bfb0e2e

```

```

SHAAlg = SHA256
Msg =
924472ca0a9d5e5c9199fb110e8e180c68d96abca675d30c55cce3da2acca0abe93e829d8553c2
9d8980d5d9b68c3378eee85c89513e1d3aa54c0f4bfbf362c28f8bd8dff17a1452620515ac8166
fe14271e183b8f8c1992babfa7ba1189183b4dd47c7a9439dde5e4cf571b5410146da24a5b8a9
87507a5bcbb10eb6b54a80
S =
3596e6a24d22280480ec71006c03d860da9dce84e9536809e2735bae083766f5fe4c352cc02eb6
72fb960e1735ec8ea557149fc5e05030ab6abb7feb0ebe8efb7a2c03f497d02453b9d2982f367e
ff3698e9b4d3f2821415b8a6687579a51ec68378c4be417a808e26323e3d4a2b330c7cef5a9c6d
55cd1bc2f250fa3c5594fdb3a98b30fb9ea3f9d319dfa3fd2c7d42ca1736858774fdcef54e3919
87a68c6a8d8a24328c9301074327608f9a6a2eb94b5d3da66a6dc912a9f4d27eb45d4fcf

[FOR 10 ITERATIONS]
.

.

[FOR 10 ITERATIONS PER SHAAlg SUPPORTED]
.

.

[mod = 2048]
[SAME FORMAT AS ABOVE]
.

.

[mod = 3072]
[SAME FORMAT AS ABOVE]
.

.

[mod = 4096]
[SAME FORMAT AS ABOVE]
.
```

B.3.3 SigVerRSA.rsp

```

# `SigVer931'
[mod = 1024]

n =
9ec4d483330916b69eee4e9b7614eafc4fbf60e74b5127a3ff5bd9d48c7ecf8418d94d1e60388b
b68546f8bc92deb1974b9def6748fb4ec93029ea8b7bea36f61c5c6aeedfd512a0f765846fad5
edacb08c3d75cf1d43b48b394c94323c3f3e9ba6612f93fe2900134217433afb088b5ca33fc4e6
b270194df077d2b6592743

SHAAlg = SHA1
e = 3
Msg =
b915e774b083e8cec80929cfbc89d87bd046f65cb43e5e78acba0380ee23794a4b17b78112bc1b
9c3254ae0c9e12aabaf62c39b063328016c39edc6106ac6bc7d76ccff67f152e05079c7dab9d85
```

```

ffaf3afa089f811a07c5e993c3571e73e5eea53bb739bf352bf391081f12818adf42e3d5ec91d5
9dfc6c67c141ca001feea7
S =
1c886e8041a0bfa57320c2033ac37eb2f8d8a96d42f3187b0f9164f37a0ce270ba35602a1e27c9
6fb6e2fdcfb25b00dalccceeb146f6a3320de97594d6de8664d3055142d408fc28c47dd380847d9
2450fad37535d366aabced070cc1fff6a6e023e2ce64e9e1914e82f384688c63beada87dd0ab71
17b5d4c1129e39b40d2440
Result = P

SHAAlg = SHA1
e = 3
Msg =
5b99290c5d6e5ba7371cdeb87551b8fe6b5d0be06d94eaf943f36bd4d707fef4310bfd18a55184
bd4be382e3b0691014cb4d02a3331ebc328f3248764d90a53f970c61b282b46ad9896b215f3bd4
b09430729db7410da075f857b2ad46cf677674e67d635c60b506d9fee1b27c5a3f85811205a601
283dcc69a9d3002a8deda3
S =
4f992d5ada1c080c144a0b0b78df1e1e74481876e5e76b9b56a0f322c4a032b0bdd31e78870067
8255e58219dad73c92809a02c5c100b87f47cffbc1ca520e82a796abf4c2f746a5ca52ab76706e
c3e633643b665da90162ca19b514457ad641ca323a2a7adac03a6d6478a7df83c91af358c5d46d
3441513a67dd6077f9f55
Result = F

SHAAlg = SHA1
e = 3
Msg =
115930d33b059329a3ac21cabd9b034fb5efb03bfc013488cc8e747620d0563e92302effa101c4
261ff1e09e23984e27a50e026e69d7c056aed41d9cb78d7b49d130e3bc5a67db836df696afe28e
82086d7450615a5a7be9762eeb29de7dea9a44b7f999a5ac3bc9e426ae608902316a95bf44b54b
fdb999c8eb67be041606b
S =
355dciae4e502821d377b78694501b4b2bcfafafa88dea405a8c6eaa4951250597ddad0da1630fd0f
2ed66fea92130664b31da24763e84759b8c7bd1f92b28525ff7b908e6cedbe914d34018c8ab7b8
2a3051793b2406bded2aab43ac9a2324961a41ade9a0049a8c2ecdf7543b79b97d1e49ed64053b
41325a403fb3a06c93543b
Result = F

SHAAlg = SHA1
e = 10001
Msg =
7261c04f0c4732cf4f78f2cfcad5b34b595aca51a798d80ff265aa300ed25d69ad5dc64edd6811
6bd2eaa790e665566cde18ac5d6f75a38701d58424b26b051a9e76b4d35ebb9949ce3317601301
1479dda65a44ea783cd5cc517cfbeb846cccd15548595291492cabb0ae85a91369bdf6e9cac20
8aea23db3f5b5dc205dba8
S =
843d6935df68e487ababc52e46d1d5ac7af0632fd8e1d86e19a3a10c3517bff8d59ba4ed09fbe4
7a1f5439cb7b63308f33d59dfe04cf64c0ecf90bd666a56b6b3a1362a47f59c1e9c6f531132c8b
4874aca35920349c68e8bf1efb0fe8571815c685134a06f54c883c819e73d8ddb48839c558f921
cf7e52783684416e993e4
Result = F

SHAAlg = SHA256
e = 3
Msg =
7c5177b2ef9ecc43c6b2048397d70f2b7dc98feabec59815aee4b49bd0a72b373fd381e94c7f3f
a6696bf74f469382e039048ceae3ef534311fcabfbe0e046932532326a0b7aa378fe8cf33ec814
e7fdfa7134278ec74113ca4f2ff468f2170cf317921d74b97f214ebc003a6781c6ec88b88f8a07
75eceea386486daf05260b
S =
1da3b0936cc9e6261e80595e46ea228c93cb7f348b2cace6a5a2704eba204b96d5cb9e29cd2cb9

```

ba968eeab994294e5f4fa2c6d44b52bc8768a802c4bc8201f267fc9e6dfa53b98677f21a77e71
78ae0166151470f628831afa59203b6a233f133544d51669eb2e5de159ed3819ef0cc504744711
6351b78ee6831e9498746
Result = F

SHAAlg = SHA256
e = 3
Msg =
7ad2f4bfbed0dba767ec7f106f4750376f2945c4c09624fbe022fe361706f8935a7252ea6f25a1
02523c5f04d847a62f92a239cef403c467b64f65367bb26ad9b1ee5d4db8f33e1946b10fc90a2a
969e8fcbb5e8464fcff447af69ffbcdd4b9cb46ed1dd0e06238560bf396494e17a5ec2f4bbcce57
aa5bfbf2beb56f55966bd0
S =
9f3e544f38658c3ab1af8a09623cb611167908c01eced7863a93d417d76098d5148485669c119a
dfbe0d7d0cda483e788c0c5b8186c192156a9a54d75d462f0da558978a7b12fe2baf9c07b3c419
1d4bf15d5f66a1c5f7079b8a535e95638a4dbf7095ef4e147b8fdd3e3498f13853710f44f778ec
6b79e95646ccb27414e4
Result = P

SHAAlg = SHA256
e = 3
Msg =
f5de826b61d81957cc4fcf26c959f1432c4b0f4f1b7fac0b685439791e77453e5961ee4b5b219b
bcdd5ced00a392f23b53a29ce8879172c3218786e6df1aa7322fcfd7f044de00b86936e29295c1
505c40e99c6c765b50762a0b1eafcc781a321e3127a34398af1318e69824c86f736e9b28f6210f
66aceb2ae8eb1c0e180708
S =
35f86e0912d099298062967bb41f6d8dadba532ecc9a66f9ad51c5dbb8de8fb29b06f8a022c4d2
8a18e7a5f9515fab51b428b7a73957fd877fcff2fe4d3a026dae0388747cb1fdbd69b20df0781b
41bfd5692f0fa4033a0399479c1a6036f8d27a9bd2018ebeb736a098090bf5ed791b9cb12cb963
ebd03dd46ffcee68b95b4b
Result = F

SHAAlg = SHA256
e = 10001
Msg =
489b177653809b9921b178eca7ddf8a31df19e45b9d40b02be551e46b5625f8efa7a9e7b7b64d7
24bd2259b12021272663b29f7c6abe59f63fb452b258c74a7f18576aee97ac2aaca6ea720e0e41
fffee196509b6543e23ba92e062cb34bdc108a819c4f830bf5cd6e5f30b2cfba748a446f2251afdf
bd2ff5bfc096b8d3ad8ed4
S =
494ce82c0af37d1b222d381b4383994f60b4897b3f6314c167bf679507436fc9f5cb6d7309d9c5
0ffe0a0838c4d2874824c78cd55a8f34654b53d9bce3989d779556b51560d92b9031ffa7f8b72d
fc6f607e829e467b17affff854ba524b6df27e53d6ef605859d2e24ebdb84db49c6af92496677d
a4d173cc054b68eb065b7f
Result = F

SHAAlg = SHA384
e = 3
Msg =
f2cdaa6531083f0f35354a7d7bead425fa5429ad0486dfcb70101310f8d4f620e01589a6d9d1a4
a524513dd1fafee7ff2c7699db07dfb1d97a7c386e7b7d687fa61c6b1a27e56664db55b6d2b2a7
7a9c58eb516f14586563f0e3a82c1a115d948a09db29e5ed8d32a16a3ee86b8e93e3c5dad9d3f4
9901701970e9b49e35de47
S =
3caed56b6f54fa617726fa4b65c90a4cc9f801cab793f20eda26e000bf118f0b5f43fc75a7ed4d
6e491a1aca3162d8529177583985f3909dd01751604b8e4124feeee7b5f556e9168b28633f9fbac
6befca8326cb8238f722cf0d05ecd7e3f26cff181556ac80bf4164e16841d4c68a81c4a1a6d13b
f5f8c70639409ce626bceb
Result = F

```

SHAAlg = SHA384
e = 3
Msg =
9b2f3ce5ef76dfd9bf3e1f916df10eea32754e0d625b7643f159b80a37fac168f094e17f877b6c
6294746027a36c473d376a5f000ec3f98ba78d1fc025579e0a7157295e96096546ce03e23e502
700421f0018449c0fc9164ea488c1d00849fc69936519e8f25574f6a03adbb1b4fe6f8ee7ac199
ba49fc305a7a6d1161aa4e
S =
4d4441be821f398047e71d418ee7fb85d50d5338b34d3e908dd371f4f2ce140bf0a102499b3e0d
5080593655327b68c568a5a22194a04ac42c8c7893e3edeecc60cf91050d0de8e3705258250a71c
40dde7beeab9ee49d736029af994638a84c35f86465a378827fedccb32b4ea6a5abdadde45ecd1
7d7353d3d0f67588aeee619
Result = F

SHAAlg = SHA384
e = 11
Msg =
e51fea92327866cac54d7a149901107e0fc1ad51e0affd4f5640b339bce2909413e3c0ad068e6d
db20978f043db901623e9dda8701fefafa9f910be3f6990ccbba1f49ef016e27290d5495c5cf8ea6
91ab817f337b2b74bfd6586e899219903887721a1c96538dc2ebc910c57e7612e6e03f315f3444
5a757b15324d1623b50085
S =
205fb3741e2886d26b181076b16fe69d527fc264372b5a4f02bd2928dbc1a53adf64f600031b7d
5baa4696025465291f84eeb33f0c68aa4c25029e104820b98030df491e8712b1638cb61836d979
c575bb4351bb1a7f8b558cb4cb6c774a3061ab1072fefef73777eccba2cf7c8c16f167ee2dff237
aed7f1b1671d48490a75b9
Result = F

SHAAlg = SHA384
e = 3
Msg =
3cd9479c6e76109d99ac8f06f227c799c261632c167e03acb774d4625c6e63751f73b3b02e428d
86c38117535617a8e511d7db3b845ee28c9a3e1d8ccb5cebea9496c11d24a979605952ecfbeab4
c2c5e6fa89e75d3d05e25ca8c7d41fda01f7838b996aba737c344267358618c6be40291396b709
e313b5e7fafac2c1b14da2
S =
3c9306f6dc9bd87a9d0bafa93025aac704d3e73ac2852c8facdba7994ac0d61f268191509713bc
6fc335259b315711df100bf662498d54373c8010da715739cdb12b09c8d959750c3a5ed7dab147
97503444f526427a96863f3ed494bafdbd955d4aa6bcba76d68f53ee23b888395d044af80273bb
4c90e9a9bfea51dc79dd2b
Result = P

SHAAlg = SHA512
e = 3
Msg =
59d5f17e6c8a8234c35c1f5c24f3c6412c8d4d1672eaf5db3c0e0594e7f30ffe0881e5f80a0249
b1d113c571db5a36c4550ddcd922f90b04bd162c791526f95bd6f2a75bfa5f19209aef54eb048
a337b0f7f5b2eabc6726abc888c3b29e0e63d2fe4d7bdfcdf031b79e1d272677a217badc237aff
09cfaac653c62dae3e72f
S =
1b20a21622cdfc81d9715cc802693b6c4982ee794fd42e3cd9d885b3a66c9abf4b8ee2254419d5
484e367bb08edbabfd36aa0b0e03d25e1b719d76a6ebcd49caf9a77826120791f9d3b759eaf07
ee8ea598f65c3c5f5a4c41b9b3af8257ff843cb0df8c0a0967406593377719bf6ba7bcb7c089c4
1c268b8e7b77d4d17a28d8
Result = F

SHAAlg = SHA512
e = 3

```

```

Msg =
cfc37b88f276536e39f3217128849ac430a1e447c8b64b64c0263536d28f58f17b502a7e094899
65f4e1fdf0714d4b8e45108b75d7bf8c8b8a07d68749f0e053646383b59aebaf2f77b85d24e0e9
c43bec0bd009f5dc0eaef556f2ff9e8e57f32076e77cee5fb94698028539adf6f96cc989b4a49
967f479819e77709fecf3b
S =
39a538fe110ac21444908d13a81d0c95d8baaecbf1ad72850956e59073ffc98ef4a78ae04151e0
c5cdaebf983e69de6643c3fef0f240851b652d5897a89e7d74bad47fd4c61c679c4d2ee1043987
2730446f7983fadbb482a7bab9ec05dda228875f2ab1bb7dae619af0d0f8b211f79ac878e4ce65
d35d5160b03fc9a1c83d11
Result = P

SHAAlg = SHA512
e = 3
Msg =
c9c7fe35a9f06697fc171a7779c9dbda5fef098dc478ca070cb846d2688ee8dec093982c78ac1
0b0c5ca1a5d38bc850a9bf509685600bcafef5e8ebaef52972a39e8b574b3ad0db1688fa9593c
ef34fb2f7fe32ac2e47d49449b96c3b4536eb21b2d49ab4522653bce2bef1f638ef05ff8ed8cb
741e9d5c58eff824b6eef1
S =
48fb17c1a361163cc23b65b5d27a09aed271a9eb6fb759c79f7cd00e5e9922887fc759a82b802e
c86dde8592c344b54b07995105ba0b1c1d9eb8234daed7162b756a04caffc0b945c008eea9f44c
9b0263e6b57246f63e79c08f6e6111e90e7617b200381fbb895ae98fdfbea60f16c9c24e8f7970
20135538efc6b8e6db3d18
Result = F

SHAAlg = SHA512
e = 10001
Msg =
80eb608f4c678f5d0de02ea11e59078d38b04f10de732b4df8f5734bbea1b5eed78f7d26c255d3
66762006584503a8cf068edaf73a3cae6d0857914ce32c28caef39802a9318f49908a9d0db024
22c4f84127e25e14e34c7ed48410840e2c534d3f398bc9a2c9eea4477d2925657e5656f1be2865
6f81694a091ba7aadefb2a
S =
30df7e92fff6ca57cee4fe6a6ed4a5ed44dc6197e6431d2e5040ce18567d6b15fdc1b40b3c0889
0c4b7312bc6456a720d17a34773c38dd4417d7d0ef4ff571910f9a8b1ccc9bcd11ac94c1cb7aa4
38625d67a1ca2cc63e9abb5340538a0c0f63ba4b4f7f1ba43498fdfe80d8d381adfd2b0acfbe4
72317197e1026bf9893db6
Result = F

n =
9ec4d483330916b69eee4e9b7614eafc4fbf60e74b5127a3ff5bd9d48c7ecf8418d94d1e60388b
b68546f8bc92deb1974b9def6748fb4ec93029ea8b7bea36f61c5c6aeedfd512a0f765846fad5
edacb08c3d75cf1d43b48b394c94323c3f3e9ba6612f93fe2900134217433afb088b5ca33fc4e6
b270194df077d2b6592743

```

[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]

```

.
.
.
.

n =
b65fd92021a7aa326a9d2234797a90b7272a3251b5a2c3b119878ab71b60016fe0070f6395019d
149b35e82d408b77a9529252d6954f5e66d649b7c4ea1704114a130e99f93357b8253a2a51ee7c
c615862904c7b958f47d2cf6343060a57764fbab6b66b25b1e8f2cebf05c1ff23fe8cccd15d0c5
f8aae94f1fcf1b1c7ad1fd

```

[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]

```
[mod = 1536]
n =
e0be7bc34325ce6c764fd6a7b09ac65a3c0d25330c978156127709050bbef1b6cc1d1473aca8d2
95606b40e6d39bc41d63294c7826c7a89eda8cf94c3fc55d2cbd172dbeeff97bba7090a0ee04c7
1034fedcc63185bc264edcea6baf007d6525322ff5cb9d709ecae8179a7e5bb7784a17cfe82810
265947c8db4da2d9f4b9c3c89d045df163f0780dd67bec9d6c0325e8ac90bd069e04db4d1ab01f
3f7797b0d33e3373911275aa4e9a986dcdd25f3950a80d03ffac15443c639148be5d04e9
.
```

B.4 Examples of *SAMPLE* Files

B.4.1 KeyGenRSA.sam

```
# CAVS 3.2
# "KeyGen RSA (X9.31)" information for "keygen1half"
# Mod sizes selected: 1024 1536 2048
# Public Keys selected:3 17 65537
# Generated on Mon May 24 15:27:21 2004

[mod = 1024]

e =
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
xp1 = 1ed3d6368e101dab9124c92ac8
xp2 = 16e5457b8844967ce83cab8c11
Xp =
b79f2c2493b4b76f329903d7555b7f5f06aaa5eaab262da1dcda8194720672a4e02229a0c71f60
aec4f0d2ed8d49ef583ca7d5eea907c10801c302acab44595
p1 = ?
p2 = ?
p = ?
xq1 = 1a5d9e3fa34fb479bedea412f6
xq2 = 1f9cca85f185341516d92e82fd
Xq =
c8387fd38fa33ddcea6a9de1b2d55410663502dbc225655a9310cceac9f4cf1bce653ec916d457
88f8113c46bc0fa42bf5e8d0c41120c1612e2ea8bb2f389eda
q1 = ?
q2 = ?
q = ?
n = ?
d = ?
```

(Same format as FAX file without the values in selected fields.)

B.4.2 SigGenRSA.sam

```
# CAVS 3.2
# "SigGen RSA (X9.31)" information for "testshas"
# Mod sizes selected: 1024 1536 2048 3072 4096
# SHA Algorithm selected:SHA1 SHA256 SHA384 SHA512
# Generated on Wed Apr 28 08:34:41 2004

[mod = 1024]

n = ?

e = ?
SHAAlg = SHA1
Msg =
940562ae51c1990882b27e7335fbb8c871db97e625d5a8f95f0f86fcbef9f27c2a2e269fc29f67
6616eeda2f9718987e4c5704fcc6475dc055478fce7c224fdda3ef665e0c354db90853fbda6b1f
fbec3dcec164b2143e425c8bb293a059c23b670ecaf115cb748bbfa98a7c9958089bad077626d3
847406dda5975412e6731d
S = ?

SHAAlg = SHA1
Msg =
61a908f8f1da17288dc06d4611df5503b79385cf80eca04ed6bbff056fedb15a7418c0bbe354b6
1d324c60a83595d2b0413eabe892a89bd2ea97227a7b8a9a64074877c346bccceeb880214099bc2
2912efbd94f9f8a51125d43249222e72e0976261b478e1b9647cd80b10d20c0f60100839c86c7b
8c0a2edcb3fc654f4e8bd9
S = ?

[FOR 10 ITERATIONS]
.
.

SHAAlg = SHA256
Msg =
dbc0695509efe8ed418c85fc9f106073638adf6e469fa35d0fcfb6161fa17f9d7223fdd537245
2a927ff7c004cf5ae8db98543b62c9a9bf914e2952de90274c553c2c60eb46edc3102d7b908380
ba6c6aa11466a2c96e20544c5b34c91f90d17f9799a57c73ca00e21d7736c42d6845382f87b7ad
a6dccaf7f51bbcfc9ac3cd0
S = ?

SHAAlg = SHA256
Msg =
26bc9efb0bd3467b5f95037fe881e3284c79d8f5237e699e4fbca84090c664bb53229f58cb0842
b0436710c9b329d98191b8f030e9c1df89b03858c1569c6ff49a7c07c4a23a8a434b0fde13be4f
94cb44ee629d5b44d336090d3de670b4f401a3d1bd85b8f085fc1e9453a4317b7cce0c2416849e
8fae6e01443ef7069659ab
S = ?

[FOR 10 ITERATIONS]
.
.

SHAAlg = SHA384
Msg =
06ac89b655b1e0bd5e7f0004cf9aa765ecc4e3c4d72b77e70ee28088358e58a91662bb28e64d28
93c9718a3d99d81892e3627aa733022d16922a28084c84c093f7e3c947b079fe03a84aa30de0e0
```

```

68623041914b8e1e54318d4d82b2247a6af5b119fb3a3d9e28b502f1919c2a1c5f7fd476fd86f
e7b4e30832d6af44d61f75
S = ?

SHAAlg = SHA384
Msg =
bc55612bb063b5f05da74569b3a39cc9abb99f2c7c93651f12c24863bca53cee258d6033851ed3
a318f9c974d098dea14778aa32f77e95bdad94ec2d3b9335c26d65c0593f6b7fee4d1c175f8bef
3772367b291a0bdda7f6b65bef1b8d471a137e25a925461061d7e45959b24e725145620a456d1f
7a42d3156079b51a8992ea
S = ?

[FOR 10 ITERATIONS]
.
.
.

SHAAlg = SHA512
Msg =
82026ab0d0445fdc66733e39a205f50a8e6e7a80619369d24982ef906760e6341ba241fa8381ab
f13ea76b52762f8b111dcc39b376486d5fd831cc37afcfcde9c3581a80bd673981e7ab6b3333dc
d43e7ff4dd77179bd7fb7be3e17b63350bd6018cbbcd86e050b3e7e60683e1f619b73b4c2e764b
74eecdbb00f4018d7ce924
S = ?

SHAAlg = SHA512
Msg =
3d52ea07b1a428f4385d4ed0fc53e8c8fed02e074fef63782492d4561d16665dff32574e2791
e8d232e7bb167052493dc33b271c032b88a6a0e002a789a195b64cce9647ebbd7ba5dbe2be3b4
87a825ffb04d16d095d70c716687d5cb7b25a886e7455c724fc9d826fda7cbe730ed9dcbb602a5
1dbd2e9c04e75c51609c1f
S = ?

[FOR 10 ITERATIONS]
.
.

[mod = 1536]

n = ?

e = ?
SHAAlg = SHA1
Msg =
9e2911b0fe05cadbbc509e8685d3bc7229ed9c8cf192ee123494d0f625214387ef8ee04e0452c7
36b156d9f1f76c2796961ca98f6aa734b2fa46a6ae4fb364a902ea06279931502e145664a8c238
37085988423046d17a9242b68196267c1e3c00c8eb0366a94d090ad8a9738f3fd50f9bb9cd382e
19a8fb326dca7853845a8a
S = ?

SHAAlg = SHA1
Msg =
36cb62c9796e73e4847cf694077aabac5aeb56ef07c665238827d26852454bc812a4910f368c1e
ddafb90af9f5542fe84c228cbf47fce7358a7fb6fc30440523cd200d3ba934a91ea0531660df2
a7895062e660520f0f95e019116c03fbcd543cbe78bc0e6d4027bdc83283563c5bb6ef132d7b30
e9256c125fab5fe7133af6
S = ?

```

```

[FOR 10 ITERATIONS]
.

.

SHAAlg = SHA256
Msg =
212850d3481b80676096014cb7c5202d98d03495b7af2360ce4027708996d2a1a09450fe42bbcd
35ba41d6484d63248661fc16bae5559ea2ec3f20a29b5f76ba46ca37b6d6d239a46e1f548b4e01
695af704a31f8fc5fd79880505507dfcdcf35cef8c62ca056d9ec357027d8146f893a011632d7f
a748cfe9b95ee1f51415a5
S = ?

SHAAlg = SHA256
Msg =
924472ca0a9d5e5c9199fb110e8e180c68d96abca675d30c55cce3da2acca0abe93e829d8553c2
9d8980d5d9b68c3378eee85c89513e1d3aa54c0f4bfb362c28f8bd8dff17a1452620515ac8166
fe14271e183b8f8c1992babffa7ba1189183b4dd47c7a9439dde5e4cfcc571b5410146da24a5b8a9
87507a5bcbb10eb6b54a80
S = ?

[FOR 10 ITERATIONS]
.

.

[FOR 10 ITERATIONS PER SHAAlg SUPPORTED]
.

.

[mod = 2048]
[SAME FORMAT AS ABOVE]
.

.

[mod = 3072]
[SAME FORMAT AS ABOVE]
.

.

[mod = 4096]
[SAME FORMAT AS ABOVE]
.
```

B.4.3 SigVerRSA.sam

```

# CAVS 3.2
# "SigVer RSA (X9.31)" information for "testshas"
# Mod sizes selected: 1024 1536
# SHA Algorithm selected: SHA1 SHA256 SHA384 SHA512
# Generated on Wed Apr 28 08:35:11 2004

[mod = 1024]
```


[FOUR SETS OF DATA FOR EACH SHAALQ SPECIFIED FOR EACH N (SEE ABOVE)]

```
.
```

```
.
```

```
.
```

```
n =  
b65fd92021a7aa326a9d2234797a90b7272a3251b5a2c3b119878ab71b60016fe0070f6395019d  
149b35e82d408b77a9529252d6954f5e66d649b7c4ea1704114a130e99f93357b8253a2a51ee7c  
c615862904c7b958f47d2cf6343060a57764fbab6b66b25b1e8f2cebf05c1ff23fe8cccd15d0c5  
f8aae94f1fcf1b1c7ad1fd
```

```
[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]
```

```
.
```

```
.
```

```
.
```

```
.
```

```
[mod = 1536]
```

```
n =  
e0be7bc34325ce6c764fd6a7b09ac65a3c0d25330c978156127709050bbef1b6cc1d1473aca8d2  
95606b40e6d39bc41d63294c7826c7a89eda8cf94c3fc55d2cbd172dbeef97bba7090a0ee04c7  
1034fedcc63185bc264edcea6baf007d6525322ff5cb9d709ecae8179a7e5bb7784a17cfe82810  
265947c8db4da2d9f4b9c3c89d045df163f0780dd67bec9d6c0325e8ac90bd069e04db4d1ab01f  
3f7797b0d33e3373911275aa4e9a986dcdd25f3950a80d03ffac15443c639148be5d04e9
```

```
[FOUR SETS OF DATA FOR EACH SHAAlg SPECIFIED FOR EACH N (SEE ABOVE)]
```

```
.
```

```
.
```

```
.
```

```
.
```